## REMARKS/ARGUMENTS

In the final Office action dated May 20, 2005, Claims 1 - 3, 14, 16 - 22, 27, 29 and 30 were rejected under 35 U.S.C. § 102. Claims 4 - 6, 9 - 12 and 24 were rejected under 35 U.S.C. § 103. Claims 7, 8, 13, 15, 23, 26, 28 and 31 were deemed allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

By this Amendment and the accompanying Request for Continued Examination, Applicants have amended the application as follows. Applicants have amended the Specification and claims 9, 10, 15, 27 and 31. Claims 14, 29 and 30 have been canceled. Claims 32 - 40 have been added. Reconsideration and reexamination are hereby requested for claims 1 - 13, 15 - 28, and 31 - 40 that are now pending in this application.

## Amendments to the Specification

Applicants have amended the Specification to correct several typographical errors. Applicants submit that no new matter has been added as the amended language is supported by the original context of the Specification.

## Response to the 35 U.S.C. §102 Rejections of the Claims

The Examiner rejected claims 1 - 3, 16 - 22, and 27 under 35 U.S.C. §102(e) as being anticipated by Silverbrook et al., U.S. Patent No. 6,334,190 (hereafter "Silverbrook"). Claims 1, 16 and 27 are independent claims.

**Claim 1**

Claim 1 is directed to an authentication engine architecture that more effectively processes multi-loop algorithms by providing specific structure that enables concurrent processing by the different loops. In particular, claim 1 recites two separate hash engines:

> a first instantiation of a multi-round authentication algorithm hash round logic in an inner hash engine;
>
> a second instantiation of a multi-round authentication algorithm hash round logic in an outer hash engine;

Moreover, claim 1 also recites specific structure that enables the multiple hash engines to operate concurrently. For example, claim 1 recites:

> a dual-frame payload data input buffer configured for loading one new data block while another data block is being processed in the inner hash engine;
>
> an initial hash state input buffer configuration for loading initial hash states to the inner and outer hash engines for concurrent inner hash and outer hash operations; and
>
> a dual-ported ROM configured for concurrent constant lookups for both inner and outer hash engines.

In contrast, Silverbrook discloses use of the standard HMAC algorithm and discloses a technique for reducing the number of registers used in a SHA1 algorithm. Silverbrook does not teach or suggest the use of concurrent multi-round hash operations as claimed.

In the final rejection, the Examiner contends that Silverbrook's "use of different registers for the first and second hash processes" teaches the use of an inner and an outer hash engine. (Emphasis added.)

Initially, Applicant respectfully submits that a register that simply stores data can not be equated to a hash engine that performs a hash operation as claimed. The Examiner's contention appears particularly untenable given the explicit "multi-round authentication algorithm hash" limitation of each hash engine.

Moreover, Applicant notes that the Examiner has not cited where Silverbrook teaches the "use of different registers for the first and second hash processes." Applicant contends that Silverbrook does not provide such a teaching.

The Examiner provides three primary citations to Silverbrook. Col.7, lines 3-5 state: "In many cases, X is broken into blocks of a particular size, and compressed over a number of rounds, with the output of one round being the input to the next." This passage refers to the rounds of a multi-round algorithm such as SHA1, MD5, etc. In accordance with conventional practice, the output of the algorithm (that is, the output of the hardware) is fed back to its input such that the same algorithm (the same hardware) repeatedly performs the same hash operation on its output data.

Col. 11, lines 9 - 27 describe the conventional HMAC algorithm. The algorithm consists of two hash function calls. Steps 1 - 4 call a hash algorithm (step 4) such as SHA1 to hash input data M. Steps 5 - 7 involve a second hash function call to the hash algorithm (step 7) to hash the result of the first hash function call.

Col. 45, lines 2 - 6 state: "Since we only deal with 2 types of messages, our padding can be constant 0s. In addition, the optimized version of the SHA1 algorithm is used, where only 16 32-bit words are used for temporary storage. These 16 registers are loaded directly by the optimized HMAC-SHA1 hardware." As will be discussed in more detail below, this passage refers to a reduction in the number of registers used in an implementation of the SHA1 algorithm.

The cited passages in columns 7 and 11 do not refer to any structure for implementing the algorithm. Hence, any disclosure of structure used to implement the algorithm may only be inferred from the written description. The two hash function calls (steps 1 - 4 and steps 5 - 7) described at column 11 are listed one after the other. All of the data used in the second loop is available once the first loop finishes. There is nothing in Silverbrook that would teach or suggest any reason to duplicate the hash engine hardware, since according to the only teaching in Silverbrook, the hash engine for the first hash function call (steps 1 - 4) would be idle during the second hash function call (steps 5 - 7). Given this description, if Silverbrook can be cited for teaching any structure at all, it must be for a single hash engine that performs the first hash function call, reloads the output data as input data, then performs the second hash function call.

The citation to Silverbrook at column, 45 likewise does not teach "use of different registers for the first and second hash processes." This passage was cited because of its reference to temporary registers.” Silverbrook discusses the purpose of the temporary registers at column 43, line 8 - column 44, line 13. Here, Silverbrook states that the "steps of the SHA1 algorithm will now be described in greater detail." Regarding the temporary registers, Silverbrook states at column 43, line 64 - column 44, line 13:

> The SHA1 Step 2 procedure is not optimized for hardware. In particular, the 80 temporary 32-bit registers use up valuable silicon on a hardware implementation. This section describes an optimization to the SHA1 algorithm that only uses 16 temporary registers. The reduction in silicon is from 2560 bits down to 512 bits, a saving of over 2000 bits. It may not be important in some applications, but in the Authentication Chip storage space must be reduced where possible. The optimization is based on the fact that although the original 16-word message block is expanded into an 80-word message block, the 80 words are not updated during the algorithm. In addition, the words rely on the previous 16 words only, and hence the expanded words can be calculated on-the-fly during processing, as long as we keep 16 words for the backward references. We require rotating counters to keep track of which register we are up to using, but the effect is to save a large amount of storage. (Emphasis added.)

From the above, it is apparent that the temporary registers are used reduce the number of register from 80 to 16 registers when implementing the SHA1 algorithm. However, SHA1 does not involve inner and outer hash processes as claimed. Rather, as discussed above, the same hardware is used in SHA1 to perform each round when multiple rounds are required (when the input data is larger than the defined block size).

Moreover, as discussed above, Silverbrook does not teach or suggest providing multiple hash engines for the two hash function calls defined by the HMAC algorithm (column 11 of Silverbrook). For the case of HMAC-SHA1, in the algorithm of column 11, the SHA1 algorithm is called at step 4 and again at step 7 ("apply H to the . . . .").

There is nothing in Silverbrook to suggest that separate hash engines are used for the two SHA1 algorithm calls at step 4 and step 7. As discussed above, if Silverbrook can be read as teaching any structure it must be for a single hash engine that sequentially performs the two hash function calls. Consequently, Silverbrook's use of temporary registers does not teach that separate registers are used for inner and outer loops.

Silverbrook also fails to teach or suggest the specific structure set forth in claim 1 that facilitates the concurrent operation of the hash engines.

Claim 1 recites, in part: "a dual-frame payload data input buffer configured for loading one new data block while another data block is being processed in the inner hash engine." (Emphasis added.) Silverbrook does use the term dual-frame buffer or any similar terms and Silverbrook does not depict any structure that is similar to the claimed dual-frame structure. This alone is grounds for traversing the rejection.

The Examiner cites the Silverbrook passages at columns 7 and 45 (quoted above) as teaching this limitation. As discussed above, the passage at column 7 merely states that some hash algorithms are multi-round algorithms. This passage says nothing regarding loading a new data block while another data block is being processed.

The Examiner states in the final Office action that column 45, lines 1 - 10 of Silverbrook teaches this limitation since it teaches the use of temporary registers. As discussed above, the passages that precede column 45 illustrate that these temporary registers are part of the SHA1 algorithm implementation. In other words, these registers are part of the hash engine. In contrast, claim 1 recites an input buffer that is separate from the hash engines.

Moreover, col. 44, lines 26 - 45 of Silverbrook illustrate that the registers are loaded separately from other operations. The sixteen registers are shown as $X_{N4}$ where $N_4$ is incremented from 0 (line 26: "$N_4$ <- 0") to 15 (line 27: "Do 16 times"). The loading of the registers is designated as "$X_{N4}$ <- InputWord $_{N4}$, etc." in rounds 0 - 4. Here, the loading of $X_{N4}$ is performed after the computations on the right hand side are performed. There is nothing in these operations that teaches or suggests that $X_{N4}$ is loaded while a data block is being processed.

New dependent claims 32 and 33 further illustrate examples of a dual-frame buffer and a multiplexer associated with a dual-frame buffer. Applicant submits that Silverbrook also fails to teach or suggest these limitations.

Claim 1 also recites, in part: "an initial hash state input buffer configuration for loading initial hash states to the inner and outer hash engines for concurrent inner hash and outer hash operations." The Examiner states in the final Office action that this limitation is met by column 45, lines 1 - 10 of Silverbrook (quoted above).

This passage does not, however, teach or suggest loading hash states for <u>concurrent</u> inner hash and outer hash operations. As discussed above, Silverbrook does not teach or suggest any concurrent hash operations. Rather, the operations in Silverbrook are performed sequentially.

Silverbrook discusses initial hash states at column 42, lines 36 - 54 and column 43, lines 29 - 30, Here, Silverbrook loads the initial hash states ($H_1$ - $H_5$) at the beginning of the SHA1 process ("Initialize the chaining variables"). See also, column 44, line 25 where Silverbrook discusses the chaining variables being loaded into the A - E registers. There is no suggestion in these passages that the variables could or should be loaded to more than one hash engine.

Moreover, column 42, lines 51 - 59 and column 43 line 64 - column 44, line 1 (cited earlier) show that Silverbrook does not load the initial hash states into the 16 temporary registers referred to in column 45 and cited by the Examiner as teaching the claimed limitation. The initial hash states are loaded into $H_1$ - $H_5$. In contrast, the 16 temporary registers take the place of the 80 temporary registers $X_{0-79}$.

Hence, Silverbrook does not teach or suggest that an initial hash state buffer may be configured to load initial hash states to two separate hash engines such that inner and outer hash operations may be performed concurrently.

New dependent claims 34 and 35 further illustrate examples of an initial hash state input buffer. Applicant submits that Silverbrook also fails to teach or suggest these limitations.

Claim 1 also recites, in part: "a dual-ported ROM configured for concurrent constant lookups for both inner and outer hash engines." Silverbrook does use the term dual-ported ROM or any similar terms and Silverbrook does not depict any structure that is similar to the claimed dual-ported ROM. This alone is grounds for traversing the rejection.

The Examiner states in the final Office action that this limitation is met by column 38 of Silverbrook. Yet the cited passage simply mentions that the system may incorporate a ROM. There is no mention of dual-port ROM. There is no mention of any interaction with two separate hash engines. Moreover, there is no teaching or suggestion that a ROM may advantageously be configured for concurrent constant lookups as claimed.

Col. 38, lines 8 - 13 of Silverbrook states: "If a given System only produces about n R-values, the sparse lookup ROM required is 10n bytes multiplied by the number of different values for M. The time taken to build the ROM depends on the amount of time enforced between calls to RD." The ROM of Silverbrook is for use with a "sparse lookup table." Column 37, line 43. In the cited passage Silverbrook is discussing the total number of lookups and the time they take but not how the lookups are conducted. Moreover, at column 37, lines 63-65, Silverbrook notes that "The attacker will therefore need to find a valid Authentication Chip and call it for each of the values in R." (Emphasis added.) This passage implies that the ROM lookups are sequential, not concurrent.

New dependent claim 36 further illustrates an example of a dual-ported ROM. Applicant submits that Silverbrook also fails to teach or suggest these limitations.

In view of the above, Applicants submit that claim 1 is not anticipated by or obvious in view of Silverbrook. Claims 2 and 3 that depend on claim 1 (and the other claims that depend on claim 1) also are patentable over Silverbrook for the reasons set forth above. In addition, these dependent claims are patentable over Silverbrook for the additional limitations that these claims contain.

**Claim 16**

Claim 16 recites, in part:

> processing the fixed-size data blocks using a multi-loop, multi-round authentication engine architecture having a hash engine core comprising an inner hash engine and an outer hash engine,
>
> pipeline hash operations of said inner hash and outer hash engines,
>
> collapse and rearrange multi-round logic to reduce rounds of hash operations.

The Examiner takes the position that the limitations quoted above are taught by col. 11 and col. 45 of Silverbrook.

However, as discussed above, Silverbrook does not teach or suggest the use of two separate hash engines.

Also as discussed above, Col. 11 merely describes the conventional HMAC algorithm. This algorithm is not inherently a pipelined operation. Silverbrook makes no other mention of pipeline operations.

Moreover, the cited passages make no mention of any logic to reduce rounds of hash operations, much less the claimed "collapse and rearrange logic." Again, only the standard HMAC algorithm has been cited. The prior Office actions are devoid of any discussion of how Silverbrook teaches this limitation.

In view of the above, the Applicants submit that Claim 16 is not anticipated by or obvious in view of Silverbrook. Claims 17 - 22 that depend on Claim 16 (and the other claims that depend on claim 16) also are patentable over Silverbrook for the reasons set forth above. In addition, these dependent claims are patentable over Silverbrook for the additional limitations that these claims contain. For example, new claim 38 recites, in part, "schedule addition computations for adding a predefined number of ending hash states of a block to initial hash states for the block in parallel with round operations for the block." This limitation is not taught or suggested by Silverbrook.

**Claim 27**

Claim 27 as amended recites, in part "schedule addition computations for adding a predefined number of ending hash states of a block to initial hash states for the block in parallel with round operations for the block." This limitation is not taught or suggested by Silverbrook. Applicant thus submits that Claim 27 and claim 28 that depends on claim 27 are not anticipated by or obvious in view of Silverbrook.

**Response to the 35 U.S.C. §103 Rejections**

The Examiner rejected claims 4 and 5 and 9 - 12 under 35 U.S.C. §103 as being unpatentable over Silverbrook in view of Sait et al., a scientific article (hereafter "Sait"). Claims 4 and 5 depend on claim 1 discussed above. Claim 9 in an independent claim. Claims 10 - 12 depend on claim 9.

Claims 6 and 24 were rejected under 35 U.S.C. §103 as being patentable over Silverbrook in view of Schneier. Claims 6 and 24 depend on claims 1 and 16, respectively, discussed above.

**Claims 9-12**

Claim 9, as amended, recites, in part, "hash round logic implementation including a plurality of addition modules each comprising, a plurality of carry save adders for computation of partial products, and a carry look-ahead adder, configured to receive at least a portion of the partial products, for computation and propagation of a final sum." Silverbrook and Sait, considered either independently or in combination, do not disclose or suggest claim 9.

Silverbrook discloses the conventional HMAC-SHA1 algorithm. In the conventional implementation of this algorithm, a variety of carry look ahead adders are used to perform addition operations for a round.

Sait discloses a technique for fast multiplications. Sait is not directed to authentication or any other form of cryptography.

The Examiner states in the initial Office action that "It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Silverbrook's system for the manipulation of secure data with Sait's technique for fast multiplication in order to process large amounts of data at high speeds." However, neither the claimed addition modules nor the HMAC-SHA1 algorithms disclosed in Silverbrook (see columns 42 - 46) involve multiplication operations. Hence, one skilled in the art looking to improve a hash engine would not have been motivated to use Sait which teaches how to improve multiplication times. Moreover, there would have been no motivation for one skilled in the art looking to improve the cryptographic techniques of Silverbrook to consider Sait because Sait is not directed to the field of cryptography.

As explained above, there is no suggestion in either reference or in the art or any motivation to combine these references. Even if the two were combined, their combination would not disclose or suggest claim 9. For example, the cited references to not disclose or suggest that an addition module may comprise "a plurality of carry save adders for computation of partial products, and a carry look-ahead adder . . . for computation and propagation of a final sum" as claimed. Applicant notes that here, the term "partial products" refers to the product (output) of the carry save adder, not to a multiplication product.

Moreover, there is no teaching or suggestion in the cited references that it may be desirable or possible to implement a hash engine using "a plurality of addition modules" each comprising a plurality of carry save adders and a carry look-ahead adder as claimed.

Applicants thus submit that Claim 9 is not obvious over Silverbrook in view of Sait. Claims 10 - 12 that depend on Claim 9 also are patentable over the cited references for the reasons set forth above. In addition, these claims are patentable over the cited references for the additional limitations that these claims contain.

## CONCLUSION

In view of the above it is submitted that the claims are patentably distinct over the cited references and that all the rejections to the claims have been overcome. Reconsideration and reexamination of the above Application is requested.

Respectfully submitted,
CHRISTIE, PARKER & HALE, LLP

By ______________________
Stephen D. Burbach
Reg. No. 40,285
626/795-9900

SDB/sdb
SDB PAS639381.1-*-08/21/05 5:32 PM